Methods for Parallel Computation of SCF NMR Chemical Shifts by GIAO Method: Efficient Integral Calculation, Multi-Fock Algorithm, and Pseudodiagonalization

KRZYSZTOF WOLINSKI, ROBERT HAACKE, JAMES F. HINTON, PETER PULAY

Department of Chemistry and Biochemistry, The University of Arkansas, Fayetteville, Arkansas 72701

Received 24 May 1996; accepted 11 August 1996

ABSTRACT: We implemented our gauge-including atomic orbital (GIAO) NMR chemical shielding program on a workstation cluster, using the parallel virtual machine (PVM) message-passing system. On a modest number of nodes, we achieved close to linear speedup. This program is characterized by several novel features. It uses the new integral program of Wolinski that calculates integrals in vectorized batches, increases efficiency, and simplifies parallelization. The self-consistent field (SCF) step includes a multi-Fock algorithm, i.e., the simultaneous calculation of several Fock matrices with the same integral set, increasing the efficiency of the direct SCF procedure. The SCF diagonalization step, which is difficult to parallelize, has been replaced by pseudo-diagonalization. The latter, widely used in semiempirical programs, becomes important in *ab initio* type calculations above a certain size, because the ultimate scaling of the diagonalization step is steeper than that of integral computation. Examples of the calculation of the NMR shieldings in large systems at the SCF level are shown. Parallelization of the density functional code is underway. © 1997 by John Wiley & Sons, Inc. *J Comput Chem* **18**: 816–825, 1997

Keywords: parallel computation; SCF NMR chemical shifts; GIAO; pseudodiagonalization

Correspondence to: P. Pulay; E-mail: pulay@uafsysb.uark.edu

Contract grant sponsors: IBM; National Science Foundation; Associated Western Univ.–Northwest Division.

Contract grant numbers: NSF: CHE-9319929; AWUND: DEFG06-89ER-75532.

CCC 0192-8651 / 97 / 060816-10

Introduction

The *ab initio* calculation of NMR shieldings has received much interest recently (see recent reviews[1–3]). Due to the gauge problem (see, e.g., refs. 1, 3), the calculation of NMR shieldings is less straightforward than the calculation of energies. Perhaps the simplest solution to the gauge problem is the use of gauge-including atomic orbitals (GIAOs)[4,5] in self-consistent field (SCF) theory, a method pioneered by Ditchfield[6] in the early seventies. The high computational cost of these early calculations precluded their wide use until 1980 when Kutzelnigg[7] developed the individual gauge for localized orbitals (IGLO) method. This method uses gauge factors on localized orbitals rather than on atomic basis functions, resulting in computational savings. A similar method, LORG, was developed by Hansen and Bouman.[8] Subsequently, it was shown[9] that the original GIAO method can also be formulated efficiently using the tools of modern analytic derivative theory[10]; the latest programs generally prefer GIAOs to other formulations.

Improvements in programs and computers have extended the applicability of GIAO chemical shift calculations to quite large molecules. One of the first large calculations was our study of a solid ice model,[11] $(H_2O)_{17}$. In an ambitious study, Pearson et al.,[12,13] using our code TX90,[9] calculated ^{13}C and ^{15}N chemical shifts in protein fragments. In conjunction with experimental shifts, these values can be used to establish the local conformation of peptide residues, yielding a much-needed new method for establishing protein structure. The only drawback of this method is the difficulty of obtaining the labeled proteins needed. Other notable examples include the calculation of chemical shifts in isomers of C_{84} by Schneider et al.,[14] using their program TURBOMOLE, the calculation of the shift tensor in sucrose by Grant and colleagues,[15] using TX90, a comprehensive study of carboranes by Bühl and Schleyer[16] and Diaz et al.,[17] and the conclusive identification of Al(I) compounds at ambient temperature by the combined theoretical–experimental study of Gauss et al.[18]

In spite of the above successes, future applications will require NMR chemical shielding calculations on still larger systems with little or no symmetry. Such calculations, when carried out on RISC workstations, become impractical because they take too much real time. Because most large calculations are interactive to some extent, long run times impede their use, even if ample computational resources are available. For example, in a recent series of calculations carried out in our laboratory on models of solid glycine, some calculations took over a week on a state of the art workstation (IBM RS6000/390). Traditional vector supercomputers do not help much. Their price/performance ratio is much worse than that of workstations, and they are not significantly faster unless the programs are fully adapted to the hardware. To give an example, at the request of a colleague we calculated the NMR shieldings of a molecule with the formula C_6H_9P using the 6–311G(d, p) basis (201 contracted Gaussians, no symmetry) in 13 h on a Pentium-90 personal computer. The same calculation, using a different program, took about 3 h on a Cray-YMP. Although this comparison has limited validity because of the different software used, it illustrates the price/performance advantage of small workstations and even PCs. Although the computing power of individual processors keeps increasing, this translates only to modest increases in the size of the largest systems that can be reasonably handled. Thus, single-processor calculations are practically limited at present to about 800 basis functions with no symmetry or 1600 with high symmetry.

The solution of the problem of NMR shift calculations for large molecules is obviously parallel processing using an array of inexpensive workstations (or PCs). NMR shieldings at the SCF or density functional level are ideally suited for this because they can be formulated in a fully direct manner,[19] i.e., without the storage of large integral files.

Method

GENERAL

Because of the complexity of modern *ab initio* (and density functional) codes, not all parts of the code can be easily parallelized. Most parallel *ab initio* codes thus use a master–slave model with replicated data structures, where less demanding computational tasks are carried out on the master and only the most steeply scaling tasks (usually integral generation and processing) are carried out in parallel. This model naturally limits the maxi-

mum number of processors that can be effectively used to about 64 processors for SCF-type calculations. Fully distributed processing, such as the ambitious project under development at the Pacific Northwest National Laboratory of the U.S. Department of Energy, aimed at solving grand challenge-type problems in quantum chemistry,[20] may allow the effective use of perhaps thousands of processors in the future but requires considerable development. In our opinion, the master–slave model with 4–20 processors offers an immediate solution for a range of current problems and is affordable to a wide circle of researchers.

Computationally, the calculation of NMR chemical shieldings at the SCF level consists of three main steps: the calculation of the SCF wave function, calculation of the GIAO derivative integrals and their contraction with the density matrix, and the solution of the coupled–perturbed SCF equations. We parallelized the computationally intensive (two-electron) part of all three steps. Parallelization of the more expensive one-electron integrals is underway. Integrals involving nuclear attraction terms, in both SCF and NMR, scale formally as N^3 and have a large prefactor.

As we envision using our software mainly on workstation clusters, the distributed memory model as implemented in version 3 of the parallel virtual machine (PVM) software[21] was adopted. The main program is changed little compared with the single processor code, except that the Fock matrix construction and analogous steps are concentrated in a separate parallel slave program.

INTEGRAL AND INTEGRAL DERIVATIVE CALCULATION

Our SCF and GIAO integral programs use a new Gaussian integral package developed primarily by K. W. at the University of Arkansas, with contributions from P. P. mainly in the design phase. This code is currently being implemented in the NWchem *ab initio* package in cooperation with Pacific Northwest National Laboratories. Details of this program will be described elsewhere (unpublished results). However, some of its features have a bearing on the parallelization strategy and will be summarized here. The basic method used is the recursive algorithm of Obara and Saika,[23] with the modifications proposed by Hamilton and Schaefer,[24] and also including some features of the program of Gill et al.[25]

The design philosophy of our integral program (which is also used for the GIAO integrals) emphasizes large systems, and this is reflected in its features summarized below.

Calculation of the integrals of identical structure in vectorized blocks (batches)[26]: Since the virtual demise of vector supercomputers, vectorization has ceased to be the catchword it used to be (somewhat undeservedly in our opinion). Nevertheless, it is still true that modern RISC processors can achieve spectacular throughput on classical vectorizable code, e.g., the basic linear algebra (BLAS) DAXPY routine (addition of constant times a vector to another vector). We have therefore tried to remove logic from the inner loops and, if possible, use BLAS constructs. In most cases, the innermost loops run over the members of a block and thus contain no logic. Although the logic of blocking similar integrals is compelling, one must adjust the block size to the hardware. Unlike vector supercomputers, RISC processors use relatively slow main memory, coupled to the CPU through a fast cache memory of limited size. Increasing the block size out of proportion to the cache memory can lead to frequent cache misses and slowdown the program. For this reason, our programs use a global variable that defines the cache size in units of 64 kB (8192 double words). The integral program then determines the appropriate block size, depending on the type of the integral. Integrals over low angular momentum functions require less cache storage and can use larger blocks. The latest generation of microprocessors is usually equipped with a more generous cache, up to several megabytes on some models, allowing larger block sizes. Integrals with the same angular momentum value and the same contraction pattern are considered of identical structure.

Blocks of integrals can be calculated in an arbitrary order.

Concentrating all logic in integer arrays: These arrays are recomputed at the start of the two-electron integral routine. The program is thus in principle open ended with respect to the angular momentum quantum number. A practical limitation to *i*-type functions is given by the one-electron integrals that are computed by an older piece of code.

Allowing general contractions[27]: These can substantially diminish the computational effort for heavier atoms. We expect that basis sets developed in the future will make increasing use of general contractions.

Allowing SP (L) type functions, i.e., shells where *s* and *p* functions share the same exponents but

not the contraction coefficients: They provide fast integral calculations for the popular basis sets.

Using only Abelian symmetry of D_{2h} or its subgroups.

A novel feature of our integral program is that it can take advantage of several identical atoms with the same basis set. This is advantageous for large molecules consisting of only a few types of atoms.

The basic unit of integral calculation is the block. Internally it may be further subdivided to subblocks to limit memory usage, but the user can address only blocks. In the parallel implementation, the master process assigns blocks to the slaves on first-come first-served basis. To diminish communication overhead, mainly latency, i.e., waiting for action, a coarse gain size is chosen in large calculations by assigning a number of blocks, say 100, to each slave at first. Because different integral blocks may require very different computing effort, this scheme, if continued to the end, may lead to poor load balance. Therefore, toward the end of calculations the granularity is decreased and single blocks are assigned to each node. This has resulted in essentially perfect load balance, to within a few tenths of a second.

SCF ITERATION: MULTI-FOCK ALGORITHM AND PSEUDODIAGONALIZATION

The direct SCF method,[19] i.e., the recalculation of two-electron integrals in each SCF cycle, has greatly extended the range of *ab initio* calculations. Although originally it was conceived as a way of substituting CPU time for disk storage, the difference density scheme[28] and careful attention to prescreening[29] made this method competitive with traditional stored integral SCF methods,[30] even in terms of CPU time. Nevertheless, the recalculation of the integrals is still the major computational expense, and it is natural to ask if it is possible to get more use out of a set of integrals than to construct a single Fock matrix. It would appear that the sequential nature of the SCF iteration precludes this. However, the DIIS method[31] used to accelerate SCF convergence can form a linear combination of a number of Fock matrices to construct an improved SCF wave function. This method can be generalized to construct several Fock matrices from a set of several density matrices, *using the same set of integrals*. Any physically reasonable density matrix could be included in the multi-Fock procedure in addition to the one used in the traditional SCF method. If used in the DIIS scheme, the inclusion of extra Fock matrices cannot harm SCF convergence. However, for an arbitrary density, one cannot expect a substantial improvement of SCF convergence. The principal problem in the multi-Fock method is thus the generation of reasonable extra density matrices. Our present scheme uses level shifts[32] to generate several (two or three) density matrices from a single Fock matrix. Both positive and small negative level shifts are used in an automatic scheme that keeps track of the HOMO−LUMO separation. The Fock matrices for all densities are evaluated simultaneously and included in a combined DIIS procedure. The latter uses a linear combination of all density matrices, in both the current SCF cycle and the previous ones, to construct an improved linear combination. This scheme greatly improves the stability of the SCF procedure early in the iteration and yields modest improvements (typically 15−20%) in the SCF cycles needed. The most likely reason for the slight improvement is that the main effect of level shifts is a general scaling of the SCF step, which is already included in the DIIS procedure. Indeed, with three or more different level shifts the density matrices are severely linearly dependent, yielding no new information. We are currently experimenting with improved methods of generating multiple density matrices, such as applying level shifts to individual orbitals that violate the Brillouin condition.

The extra CPU time needed for constructing additional Fock matrices is very low on our workstation cluster, although the construction of a single Fock matrix is quite expensive (up to 50% of the integral evaluation time for lightly contracted basis sets). The high expense of a single Fock matrix construction is probably due to the indirect addressing that is necessary if the integrals are calculated in batches. Address calculation does not have to be repeated if a few Fock matrices are calculated simultaneously, and thus the extra computational work for additional Fock matrices is small. The major drawback of the multi-Fock method is the increased memory demand.

A notorious problem in parallel SCF calculations is the diagonalization of the Fock matrix. The usual matrix diagonalization programs, e.g., Householder reduction[33] to tridiagonal form, followed by the QR algorithm, have proved difficult to parallelize. The formal scaling of this step is $O(n^3)$, compared to $O(n^4)$ for integrals. (The pro-

gram we use, SDIAG2,[34] scales even worse than n^3 for dimensions over 500, presumably because of paging problems.) For large molecules integrals scale effectively only as $O(n^{2.5})$ or so, and in the limit[35] only as $O(n^2)$. This suggests that ultimately diagonalization will dominate over integral calculation, just as in semiempirical calculations. Modern semiempirical codes use an alternative method, pseudodiagonalization.[36] It is quite natural that pseudodiagonalization was first developed for semiempirical calculations, because diagonalization constitutes the bulk of the computational effort in these methods. The essential idea of pseudodiagonalization is that the accurate diagonalization of the Fock matrix is not necessary during SCF iteration, for two reasons. First, it is not necessary to canonicalize the orbitals during SCF iteration. Therefore, no orbital rotations need to be carried out between two occupied orbitals or two virtual ones. However, for performing orbital rotations between occupied and virtual orbitals, it is important to have approximately canonical orbitals. Second, the annihilation of the occupied–virtual elements of the Fock matrix, i.e., the elements violating the Brillouin condition, does not have to be exact. It suffices to diminish them, e.g., through an orbital transformation that is accurate to first order of perturbation theory only. The reason for this is that two-electron contribution to the Fock matrix, which are not considered in the usual SCF procedure based on diagonalization of the Fock matrix, causes an error that is larger than the error arising from diagonalization, which is accurate only in first order. After a few initial steps, when the Fock matrix becomes roughly diagonal in orbital basis and reasonable orbital energies are available, exact diagonalization is replaced by a series of Jacobi rotations between occupied and virtual orbitals. Moreover, these rotations have to be carried out only for off-diagonal matrix elements that exceed a certain fraction, say 0.005 times, of the maximum rotation angle. Pseudorotation requires substantially fewer operations than diagonalization and it also vectorizes better, diminishing the serial overhead substantially. For a large number of processors it will be important to parallelize this step as well. Because it consists only of matrix multiplications and DAXPY routines, it is much easier to parallelize than finite matrix diagonalization methods, although we have not yet implemented this step. Shepard[37] recently described a method based on the second-order SCF procedure of Bacskay,[38] which also avoids the solving of the secular equation and the canonicalization of the orbitals. However, as the present work shows, pseudodiagonalization familiar from the semiempirical field[36] can be equally useful in *ab initio* calculations over a certain size.

SCF AND COUPLED-PERTURBED HARTREE-FOCK (CPHF) ITERATION: DATA STORAGE AND COMMUNICATION

Following Clementi et al.,[39] our present implementation uses replicated data structures for the Fock and density matrices and for the analogous structures in other program parts. This method has been generalized to related computational tasks[40,41] and extended to shared-memory architectures.[42] However, while simple, such a strategy does not scale properly for a large number of processors because memory requirements and communication overhead grows without limit. For this reason Colvin et al.,[43] Furlani and King,[44] and Wong and Harrison[20] recently developed techniques for distributed data storage in SCF calculations. Unfortunately, it is difficult to reconcile maximum efficiency in the storage of the distributed Fock matrix with maximum efficiency of the integral calculation, because they require a different ordering of the basis functions.

The following analysis suggests that on a workstation cluster using the present generation of CPUs and communication hardware, the problem of replicated data structures may not be as serious as generally assumed. To be definite, we consider a representative calculation with no symmetry and about 2000 basis functions. The memory requirement for the storage of a single density Fock matrix pair is about 4 MW (4 million double words) or 32 MB. This may be a significant problem on systems with thin nodes, i.e., nodes without disk-based virtual memory and only a limited amount of fast memory. However, a workstation cluster is usually able to accommodate much larger data structures in virtual memory. Those parts of the Fock and density matrices that are not actually used on a given node are paged out and reside on the disk in virtual memory. A second objection to replicated data structures is the communication overhead, particularly the global summation of the Fock matrix contributions. When implemented on a simple Ethernet loop as in our system, this requires the sequential transmission of P Fock matrices where P is the number of processors. Taking our previous example and 31 processors, and as-

suming an effective transmission speed of 1 MB/s on ordinary Ethernet, this takes $16 \times 31 \approx 500$ s, which is a significant amount of time, although probably still not large compared to integral computation. In the CPHF scheme for NMR shieldings, three derivative Fock matrices must be transmitted, and communication overhead becomes a more significant concern; the same is true for the multi-Fock method described in an earlier section. However, communication times are reduced by an order of magnitude, to 50 s, if the new fast Ethernet (100 Mbits/s $\approx$ 10 Mbyte/s) is used. Moreover, by equipping roughly half of the nodes with two communication boards, or using an Ethernet switch, global summation can be carried out in $[\log_2(P + 1) - 1] \times T$ time, where T is the time to transmit one Fock matrix. This is illustrated in Figure 1. In our example, T on the fast Ethernet is about 1.6 s, and the total is only $\approx$ 13 s, not counting latencies. This is quite tolerable in a calculation this large; even with 128 nodes it would be reasonable. For the limited number of processors used in this study, communication overhead was not a significant problem, in spite of using only a simple loop of ordinary (10 MB/s) Ethernet cable for communication. Latencies, while quite significant for transmitting a large number of small blocks, are probably not important in the present case where relatively few large blocks are transmitted.

NMR SHIFT CALCULATION

There are two main computational steps in the GIAO method for NMR shielding constants: the evaluation of the GIAO derivative two-electron integrals (g^1) and their contraction with the zeroth-order density matrix (D^0) elements, and the iterative solution of the CPHF equations.

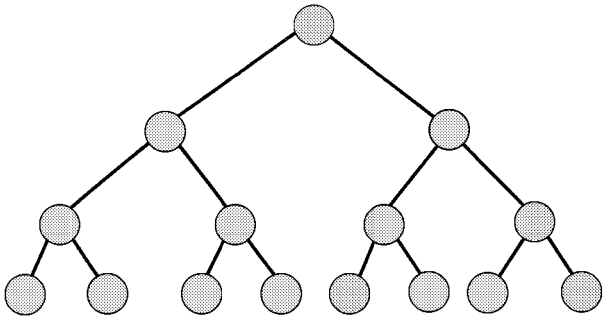

FIGURE 1. Scheme of a binary communication tree allowing the global summation of the Fock matrix in time proportional to $[\log_2(P + 1) - 1]$ where P is the number of the processors, $P = 15$ in this example. Two communication cards are used per internal node, and one card for each terminal node.

The first step has to be performed only once[9] and appropriate parts of the resulting first-order Fock matrices $F(D^0, g^1)$ are kept in memory. The solution of the CPHF equations requires recalculation of the remaining parts of the first-order Fock matrices, i.e., $F(D^1, g^0)$ in each iteration. These two steps were parallelized.

The CPHF step has been implemented in a manner analogous to the SCF procedure. We use two thresholds for integral evaluation. For the early iterations a loose integral threshold (usually 10^{-7}) is used. When the first-order density reaches a given level of convergence (say $0.5 * 10^{-3}$) or after a few iterations in cases with convergence problems, the integral threshold is sharpened. The program switches to the final threshold (usually 10^{-10}) as the desired convergence is approached, and thereafter full integral evaluation is performed. This approach results in about 15% saving in CPU time, compared to using the same (final) integral threshold throughout. Additional speedup is achieved by solving the CPHF equations for *changes* in the first-order density matrix instead of for the full matrix. This yields a higher efficiency in the integral prescreening procedure, just as it does in direct SCF.[18,45] This technique alone brings almost a 30% saving in the CPHF step. Combining the loose integral threshold and ''delta density'' approaches, we were able to speed up the CPHF solution by a factor of almost two, in some cases even more.

The bulk of the time needed for NMR shieldings is used to determine the first-order perturbed wave function and is needed for a single nucleus already. For each additional nucleus, the nuclear magnetic dipole integrals have to be calculated and contracted with the first-order density. For a large number of nuclei, this work is not negligible and ought to be parallelized.

The GIAO calculation of NMR shielding, using density functional theory (DFT) without current terms,[46,47] the ''uncoupled DFT'' approach,[48] does not require a CPHF step because the first-order response vanishes. This save considerable computer time. On the other hand, the numerical integration in the SCF program is quite demanding if a dense grid is used. Parallelization of our DFT NMR shielding program is in progress.

Calculations, Timings, and Discussion

In this section we demonstrate the performance of our parallel TX95/SCF/GIAO program as implemented on a cluster of six IBM RS6000/390 workstations. Table I presents timing results for five large (on the *ab initio* scale) molecular systems recently studied in our group. Molecular symmetry was not used in these calculations, mainly because it has not yet been implemented in TX95. However, symmetry is probably not important for the projected applications to large systems. We describe briefly the molecules and basis sets used and the rationale for the calculations. The results and comparisons with the experiment will be published separately; only the timings are discussed here.

We recently calculated the *ab initio* vibrational[49,50] and NMR[51] spectra of porphyrin and magnesium porphyrin (MgP), including the MgP—H_2O complex. Dodecamethyl-cyclohexasilane, $(Me_2Si)_6$, was taken from a recent study of the ^{29}Si magnetic shielding anisotropy of silicon rings of different sizes.[52] We are presently performing a computational study of the NMR shielding tensors in solid-state glycine, modeled by finite clusters of glycine molecules. Timings for $(glycine)_6$ are shown. The structure of gramicidin A (276 atoms) was recently determined in a comprehensive experimental study conducted in the NMR laboratory of (J. F. H.). Timings for the chemical shift calculations for an 82-atom fragment of this molecule are shown.

For porphyrin and magnesium porphyrin, geometries optimized at the B3-LYP/6–31G* level were used.[34,35] The SCF/3–21G* optimized geometry for dodecamethyl-cyclohexasilane was taken from ref. 52. The atomic coordinates of the glycine cluster were taken from a neutron diffraction study of solid glycine.[53] The cluster consists of two central molecules surrounded by four others; all hydrogen bonds found in the crystal are present in our model.

We used different basis sets, starting with the 6–31G* polarized split-valence set, which is proba-

TABLE I.
Performance of TX95 / GIAO Program for NMR Shielding Tensor Calculations on Cluster of Six IBM / RISC / 390 Workstations.

System and Size	Step	No. Iterations	CPU Time Master	CPU Time Slaves	Elapsed Time Master	Elapsed Time Slaves	Efficiency[a]	Total Time[b]
1. Porphine: $C_{20}N_4H_{14}$, 6–311G(d, p) basis set								
38 atoms	SCF	5	9	474	100	96	0.82	439
516 CGTO	NMR	11	19	1178	324	322	0.61	
2. Mg—H_2O—Porphine: Mg—H_2O—$C_{20}N_4H_{12}$, 6–311G(d, p) / McLean–Chandler basis set								
40 atoms	SCF	10	15	3303	628	621	0.89	1379
566 CGTO	NMR	10	23	3813	723	720	0.88	
3. Dodecamethyl-cyclohexasilane: $Si_6C_{12}H_{36}$, Dunning's gen. contr. DZ basis set								
54 atoms	SCF	5	8	743	140	135	0.92	534
456 CGTO	NMR	9	23	1949	375	371	0.88	
4. Cluster $(glycine)_6$: $(C_2O_2NH_5)_6$, 6–31G* basis set								
60 atoms	SCF	6	11	400	90	84	0.80	307
510 CGTO	NMR	10	23	763	191	190	0.67	
5. Cluster $(glycine)_6$: $(C_2O_2NH_5)_6$, 6–311G(d, p) basis set								
60 atoms	SCF	4	47	1119	267	248	0.75	937
720 CGTO	NMR	10	50	2792	628	624	0.75	
6. Gramicidin fragment: $C_{25}O_7N_6H_{44}$, 6–31G* basis set								
82 atoms	SCF	7	43	1558	353	329	0.79	1014
658 CGTO	NMR	13	55	2285	578	575	0.66	

Values are in minutes. Performance is without symmetry. See text for references and the source of molecular geometry.
[a]Ratio of the cumulative CPU time and the elapsed time of the slave processors, divided by their number.
[b]NMR time denotes the time needed to calculate NMR shieldings for a single nucleus. Total time includes the time needed to calculate shieldings for additional nuclei (see text).

bly the minimum useful basis for NMR shift calculations. This basis set was used in calculations for the gramicidin fragment and for one of the glycine cluster calculations. The porphyrin, magnesium porphyrin-H_2O, and the larger $(glycine)_6$ calculations employed the polarized triple-zeta 6−311G(d, p) basis set (with McLean and Chandler[54] for Mg). For dodecamethyl-cyclohexasilane we used the recent correlation-consistent set of Dunning.[55] This is a generally contracted, polarized double-zeta basis.

Overall, the molecular systems considered in this study contained from 38 to 82 atoms and basis sets comprising 456−720 contracted Gaussian-type orbitals (CGTO).

The longest run (Table I) was for magnesium porphyrin-H_2O, that was completed in 23 h elapsed time, which is quite reasonable for a system of this size without symmetry (40 atoms and 566 basis functions). Only about 5 h were needed for 60 atoms/510 basis functions in the case of $(glycine)_6$ using the 6−31G* basis set. However, the timings for the SCF step are somewhat optimistic because preconverged wave functions were used as a starting guess for all SCF calculations. These starting vectors were obtained from separate runs using smaller basis sets without polarization functions.

The results in Table I show that one CPHF iteration takes, on average, only 1.35 times longer than one SCF iteration. Assuming that about 12 iterations for the SCF and 10 for the CPHF will be needed on average for GIAO/NMR calculations, we may expect that the NMR/SCF time ratio for large systems will be close to 1 as it is in the IGLO method.[1]

The efficiency of our parallel implementation is also shown in Table I. We define this quantity as a ratio between the total CPU and elapsed times of the slave processes, divided by the number of processors (six in the present case). Ideally this value should approximate 1 but will not quite attain it due to unavoidable wait periods. Note that efficiencies lower than 1 represent a loss only if the slave CPUs are idle during the wait periods. Background jobs running on these processors can utilize the wait periods, suggesting that the most efficient strategy for processor utilization is a mix of parallel and background jobs, even if it results in longer elapsed times. The efficiency of the SCF step (average 0.83) is higher in our examples than for the NMR one (average 0.74). The probable reason for this is that we have not yet implemented our variable granularity load balancing algorithm in the NMR step.

To make a direct estimate of the speedup achieved due to the parallel computing, we performed calculations for dodecamethyl-cyclohexasilane using only one processor. For the same number of SCF and CPHF iterations as in the case of six processors (see Table I), the elapsed time for SCF and NMR was 777 and 2033 min, respectively. The whole one-processor run was completed in 2810 min, i.e., it took 5.3 times longer than a six-processor one.

It is of interest to consider the scaling of the calculations with the number of the basis functions. These data are collected in Table II for a single-processor run. Ultimately, the times should scale with N^2 at constant basis set quality.[35] However, this favorable scaling is attained only when the average distance between the atoms becomes significantly larger than the average diameter of the most diffuse basis function (the inverse square root of its exponent). This condition is quickly satisfied for small basis sets, e.g., 3−21G, and essentially linear (1-dimensional) molecules. For basis sets containing diffuse functions, and for 3-dimensional molecules, like the glycine cluster, the threshold for N^2 scaling is reached later: for

TABLE II.
Scaling Properties of TX95 / GIAO Program for NMR Shielding Tensors.

System	Basis Set Dimension	SCF	GIAO (2 Element)	CPHF	NMR Total	Total Time
$(Glycine)_4$	480	454	286	645	938	1404
$(Glycine)_6$	720	1475	783	2148	2952	4446
Scaling n^x	x	2.91	2.48	2.97	2.83	2.84

Timings in minutes on one IBM / RISC / 390 workstation. Basis set 6−311G(d, p); 4 SCF and 10 CPHF iterations.

the $(\text{glycine})_n$ clusters ($n = 4-6$) using the 6−311G(d, p) basis we obtain an $N^{2.8}$ scaling. However, it is expected that the scaling improves for larger molecules. Even under the pessimistic assumption that the scaling remains $N^{2.8}$, we should be able to perform SCF/NMR calculations for about 120 atoms and 1500 basis functions (about twice the size of the larger glycine calculation) in about 5 days on our six-processor cluster.

The authors will make the program available in due course.

Acknowledgments

This project was supported by IBM Co. by a grant to P. P. and J. F. H. and by the National Science Foundation under Grant CHE-9319929 to P. P. K. W. acknowledges support by Associated Western Universities−Northwest Division under Grant DEFG06-89ER-75522 with the U.S. Department of Energy. We also thank the National Science Foundation and IBM Co. for an equipment grant to P. P. and J. F. H. We are also grateful to Dr. R. A. Kendall for valuable comments.

References

1. W. Kutzelnigg, U. Fleischer, and M. Schindler, In *NMR—Basic Principles and Progress*, Vol. 23, Springer, Berlin, 1990, p. 165.
2. C. J. Jameson, *Nuclear Magn. Reson.*, **21,** 36 (1992), **22,** 59 (1993).
3. P. Pulay and J. F. Hinton, In *Encyclopaedia of Magnetic Resonance*, D. Grant and R. K. Harris, Eds., Wiley, New York, 1995.
4. F. London, *J. Phys. Radium*, **8,** 397 (1937).
5. H. F. Hameka, *Mol. Phys.*, **1,** 203 (1958).
6. R. Ditchfield, *Mol. Phys.*, **27,** 789 (1974).
7. (a) W. Kutzelnigg, *Isr. J. Chem.*, **19,** 980 (1981); (b) M. Schindler and W. Kutzelnigg, *J. Chem. Phys.*, **76,** 1919 (1982).
8. A. E. Hansen and T. D. Bouman, *J. Chem. Phys.*, **82,** 5035 (1985).
9. K. Wolinski, J. F. Hinton, and P. Pulay, *J. Am. Chem. Soc.*, **112,** 8251 (1990).
10. P. Pulay, *Adv. Chem. Phys.*, **69,** 241 (1987).
11. J. F. Hinton, P. Guthrie, P. Pulay, and K. Wolinski, *J. Am. Chem. Soc.*, **114,** 1604 (1992).
12. A. C. de Dios, J. G. Pearson, and E. Oldfield, *Science*, **260,** 1491 (1993).
13. J. G. Pearson, J.-F. Wang, J. L. Markley, H.-B. Le, and E. Oldfield, *J. Am. Chem. Soc.*, **117,** 8823 (1995).
14. U. Schneider, S. Richard, M. M. Kappes, and R. Ahlrichs, *Chem. Phys. Lett.*, **210,** 165 (1993).
15. D. M. Grant, J. C. Facelli, D. W. Alderman, and M. H. Sherwood, In *Nuclear Magnetic Shieldings and Molecular Structure*, Vol. 386, J. A. Tossell, Ed., Kluwer Academic Publishers, Dordrecht, 1993, p. 367.
16. M. Bühl and P. V. R. Schleyer, *J. Am. Chem. Soc.*, **114,** 477 (1992).
17. M. Diaz, J. Jaballas, J. Arias, H. Lee, and T. Onak, *J. Am. Chem. Soc.*, **118,** 405 (1996).
18. J. Gauss, U. Schneider, R. Ahlrichs, C. Dohmeier, and H. Schnöckel, *J. Amer. Chem. Soc.*, **115,** 2402 (1993).
19. J. Almlöf, K. Faegri, Jr., and K. Korsell, *J. Comp. Chem.*, **3,** 385 (1982).
20. A. T. Wong and R. J. Harrison, *J. Comp. Chem.*, **16,** 1291 (1995).
21. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine*, MIT Press, Cambridge, MA, 1994.
22. K. Wolinski and P. Pulay, unpublished results.
23. S. Obara and A. Saika, *J. Chem. Phys.*, **84,** 3963 (1985).
24. T. P. Hamilton and H. F. Schaefer, *Chem. Phys.*, **150,** 163 (1991).
25. P. M. W. Gill, M. Head−Gordon, and J. A. Pople, *Int. J. Quantum Chem. Quant. Chem. Symp.*, **23,** 269 (1989).
26. V. R. Saunders and M. F. Guest, *Comp. Phys. Commun.*, **26,** 389 (1982).
27. R. C. Raffenetti, *J. Chem. Phys.*, **58,** 4452 (1973).
28. M. Häser, J. Almlöf, and M. W. Feyereisen, *Theor. Chim. Acta*, **79,** 115 (1991).
29. H. Horn, H. Weiss, M. Häser, M. Ehrig, and R. Ahlrichs, *J. Comp. Chem.*, **12,** 1058 (1991).
30. H. B. Schlegel and M. J. Frisch, In *Theoretical and Computational Methods for Organic Chemistry*, NATO-ASI Series C339, J. S. Formosinho, I. G. Csizmadia, and L. G. Arnaut, Eds., Kluwer, Dordrecht, The Netherlands, 1991, p. 5.
31. (a) P. Pulay, *Chem. Phys. Lett.*, **73,** 393 (1980); (b) P. Pulay, *J. Comp. Chem.*, **3,** 556 (1982).
32. V. R. Saunders and I. H. Hillier, *Int. J. Quantum Chem.*, **7,** 699 (1973).
33. See, e.g. B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice−Hall, Englewood Cliffs, NJ, 1980.
34. SDIAG2, Leibniz-Rechenzentrum, Munich, 1966.
35. V. Dyczmonds, *Theor. Chim. Acta*, **28,** 307 (1973).
36. J. Stewart, P. Császár, and P. Pulay, *J. Comp. Chem.*, **3,** 227 (1982).
37. R. Shepard, *Theor. Chim. Acta*, **84,** 343 (1993).
38. G. B. Bacskay, *Chem. Phys.*, **61,** 385 (1981).
39. E. Clementi, G. Corongiu, J. Detrich, S. Chin, and L. Domingo, *Int. J. Quant. Chem. Quant. Chem. Symp.*, **18,** 601 (1984).
40. M. Dupuis and J. D. Watts, *Theor. Chim. Acta*, **71,** 91 (1987).
41. M. F. Guest, R. J. Harrison, J. H. van Lenthe, and L. C. H. van Corler, *Theor. Chim. Acta*, **71,** 117 (1987).
42. R. J. Harrison and R. A. Kendall, *Theor. Chim. Acta*, **79,** 337 (1991).
43. M. E. Colvin, C. L. Janssen, R. A. Whiteside, and C. H. Tong, *Theor. Chim. Acta*, **84,** 301 (1993).

44. T. R. Furlani and H. F. King, *J. Comp. Chem.*, **16,** 91 (1995).
45. M. Häser and R. Ahlrichs, *J. Comp. Chem.*, **10,** 104 (1989).
46. G. Schreckenbach and T. Ziegler, *J. Phys. Chem.*, **99,** 606 (1995).
47. G. Rauhut, S. Puyear, K. Wolinski, and P. Pulay, *J. Phys. Chem.*, **100,** 6310 (1996).
48. V. G. Malkin, O. L. Malkina, and D. R. Salahub, *Chem. Phys. Lett.*, **204,** 80 (1993).
49. P. M. Kozlowski, M. Z. Zgierski, and P. Pulay, *Chem. Phys. Lett.*, **247,** 379 (1995).
50. P. M. Kozlowski, A. A. Jarzecki, and P. Pulay, *J. Phys. Chem.*, in press (1996).
51. P. M. Kozlowski, K. Wolinski, P. Pulay, B.-H. Ye, X.-Y. Li, *J. Am. Chem. Soc.*, submitted.
52. G. Magyarfalvi and P. Pulay, *Chem. Phys. Lett.*, **241,** 393 (1995).
53. P. G. Josson and A. Kvick, *Acta Cryst. B*, **28,** 1827 (1972).
54. A. D. McLean and G. S. Chandler, *J. Chem. Phys.*, **72,** 5639 (1980).
55. T. H. Dunning, *J. Chem. Phys.*, **90,** 1007 (1989).